IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Appellant(s):	Elmootazbellah N. Elnozahy et al.		
Assignee:	International Business Machines Corp.		
Title:	HARDWARE SUPPORT FOR SUPERPAGE COALESCING		
Serial No.:	10/713,733	Filing Date:	November 13, 2003
Examiner:	A. Savla	Group Art Unit:	2185
Docket No.:	AUS920030760US1		

Austin, Texas
November 5, 2008

COMMISSIONER FOR PATENTS
Mail Stop Appeal Brief—Patents
P.O. Box 1450
Alexandria, VA 22313-1450
—filed electronically via EFS-Web—

APPEAL BRIEF UNDER 37 C.F.R. §41.37

Dear Sir:

This Appeal Brief is submitted pursuant to the Notice of Appeal filed September 7, 2008. Appellants have previously paid $510.00 for the submission of the prior Appeal Brief in this case dated March 5, 2008. The revised fee under 37 C.F.R. §41.20(b)(2) is $540.00, so please charge the difference of $30.00, and any additional required fees, to deposit account number 09-0447.

STATEMENT OF THE REAL PARTY IN INTEREST

The real party in interest to this appeal is International Business Machines Corporation, the assignee of record.

STATEMENT OF RELATED CASES

There are no other prior or pending appeals, interferences or judicial proceedings known to Appellants or their legal representative which are related to, directly affect or would be directly affected by or have a bearing on the Board's decision in this appeal.

JURISDICTIONAL STATEMENT

The Board has jurisdiction to consider this appeal under 35 U.S.C. §134(a). The Office Action setting out the rejection from which this appeal is taken has a mailing date of June 12, 2008. Appellants filed a Notice of Appeal on September 7, 2008. This Appeal Brief is being filed on November 5, 2008.

TABLE OF CONTENTS

TABLE OF AUTHORITIES

STATUS OF AMENDMENTS

There have been no amendments filed subsequent to the rejection that is the subject of this appeal.

GROUNDS OF REJECTION TO BE REVIEWED

Claims 1-5, 7, 9-11, 14, 17 and 19-20 were rejected under 35 U.S.C. §103(a) as being unpatentable over Applicants' admitted prior art (AAPA) in view of U.S. Patent No. 6,434,681 (Armangau).

Claims 6 and 16 were rejected under §103(a) as being unpatentable over AAPA in view of Armangau and U.S. Patent No. 6,175,906 (Christie).

Claim 8 was rejected under §103(a) as being unpatentable over AAPA in view of Armangau and the article "Reducing TLB and Memory Overhead Using Online Superpage Promotion" (Romer).

Claims 12-13 were rejected under §103(a) as being unpatentable over AAPA in view of Armangau and U.S. Patent No. 6,725,289 (Waldspurger).

Claim 15 was rejected under §103(a) as being unpatentable over AAPA in view of Armangau and the article "Surpassing the TLB Performance of Superpages with Less Operating System Support" (Talluri).

Claim 18 was rejected under §103(a) as being unpatentable over AAPA in view of Armangau and U.S. Patent No. 5,974,507 (Arimilli).

STATEMENT OF FACTS

The material facts relevant to the rejections on appeal are as follows. The Office Action asserts that a "processor" is analogous to a "memory controller" (¶3, p. 3). Appellants' specification distinguishes between a processor and a memory controller (p. 1, ll. 11-24; p. 11, ll. 20-25; Fig. 4, ref. numerals 56, 42). A memory controller is responsive to read and write instructions from a bus connected to the processor (specification p. 10, l. 21; p. 11, ll. 22-24). In rejecting Claim 1 the Office Action relies on page 5, lines 16-17 of the specification. That text states that the operating system (OS) "uses the processor to copy data …" (emphasis added). Claim 1 recites a "memory controller." Claim 1 does not recite a "processor". The Office Action even distinguishes the processing unit of Armangau from a memory controller (¶12, p. 7).

In rejecting Claim 1 the Office Action (¶3, p. 3) also relies on the description at page 5, lines 13-16 of Appellants' specification. The specification notes at page 5, lines 11-12, that this prior art "solution resorts to software-directed memory copying." In contrast the present invention is hardware-based (specification p. 10, l. 12). The present invention reduces or eliminates poor translation-lookaside buffer (TLB) behavior that occurs with the prior art software-directed solution (specification p. 10, ll. 13-14). The specific hardware that supports the invention is

the memory controller recited in the claims. The memory controller receives simplified instructions which define the set of pages to be copied (specification p. 10, ll. 25-27). A state engine within the memory controller uses direct memory access to carry out copying (p. 11, ll. 7-10). The OS can accordingly use the new mappings right away without waiting for physical copying to complete (p. 11, ll. 14-15).

The Office Action acknowledges that AAPA does not disclose accessing the virtual superpage using the new mapping while the memory controller is copying pages (¶8, p. 5). The Office Action further acknowledges that AAPA does not disclose using a mapping table for this function (id.). The Office Action asserts that Armangau accesses a virtual superpage using a new mapping while the memory controller is still copying pages (id.). However, Armangau never discusses page migration or the creation of a virtual superpage. In Armangau, if any process is required to access data it does so using the production volume and not the snapshot copy (col. 2, ll. 16-18).

In rejecting Claims 3 and 9 the Office Action (¶5, p. 4; ¶9, p. 6) refers solely to Armangau column 2, lines 16-18. That text of Armangau says only that the production set is accessible during maintenance of the snapshot copy.

In rejecting Claims 6 and 16 the Office Action (¶17, p. 10; ¶18, p. 11) refers to Christie column 4, lines 40-50. That text of Christie refers to "virtual tags"

which are invalidated by deasserting "valid bits." The valid bits are in a "valid register" (ref. numeral 114 of Fig. 1). Christie distinguishes between these valid bits and virtual addresses (col. 3, l. 61 through col. 4, l. 2). The virtual addresses are stored in the virtual tag register (col. 3, ll. 60-63; ref. numeral 110 of Fig. 1). Claims 6 and 16 do not recite the deassertion of a valid bit. Claims 6 and 16 recite modification of an address tag according to the new page mapping. This feature of Appellants' invention allows the superpage construction to complete without having to flush the contents of any affected cache memory (specification p. 12, ll. 16-22).

In rejecting Claims 12 and 13 the Office Action asserts that the mapping module of Waldspurger is analogous to the recited state engine (¶22, p. 12). The mapping module of Waldspurger is a component of the operating system (col. 6, ll. 1-4). The mapping module is accordingly software, not hardware (col. 4, ll. 55-56). The mapping module (ref. numeral 610 of Waldspurger) is described and illustrated as being part of a manager (ref. numeral 605; col. 6, ll. 57-59). The manager is in the intermediate software layer described by Waldspurger (col. 4, ll. 15-16; col. 8, ll. 15-17). The state machine recited in Claims 12 and 13 (ref. numeral 48 in Appellants' Fig. 3) is part of the memory subsystem of the present invention (ref. numeral 40). This memory subsystem is hardware (specification p. 10, l. 12). In rejecting Claim 13, the Office Action asserts that the memory

management unit (MMU) of Waldspurger is analogous to a DMA engine (¶23, p. 13). Claim 13 specifies that the DMA engine carries out actual copying of the memory pages. In addressing this recitation the Office Action (¶23, p. 13) relies on Waldspurger column 7, line 67 through column 8, line 3. This text states in part that "MMU 116 then performs the actual translation of VPNs to PPNs using these mappings." This text does not state that the MMU performs "copying." Translation of a virtual page number to a physical page number does not involve copying of the page. The use of a state engine with a DMA engine as recited in Claim 13 allows copying to be gradually completed (p. 11, ll. 7-14).

The Office Action acknowledges that the combination of AAPA and Armangau does not disclose updating TLB entries prior to completion of copying of the memory pages (¶25, p. 14). The Office Action asserts that Talluri teaches such updating of the TLB entries (id.). In rejecting Claim 15 the Office Action relies on page 3 of Talluri, right column, second full paragraph (id.). That text of Talluri first describes copying memory pages, and then updating the TLBs. However, Talluri never states that the TLB entries are updated prior to completion of copying.

The Office Action acknowledges that the combination of AAPA and Armangau does not disclose relocating a cache entry based on a changed congruence class for a modified address tag (¶27, p. 15). In rejecting Claim 18 the

Office Action relies on column 6, lines 43-66 of Arimilli. That text of Arimilli describes the arbitrary assignment of addresses to congruence classes by switching address bits. However, Arimilli never states that a cache entry is relocated.

ARGUMENT

Appellants would respectfully submit that the following points represent errors in the Office Action rejection dated June 12, 2008, of Claims 1-20.

Rejection of Claims 1-5, 7, 9-11, 14, 17 and 19-20 under §103(a)

The proposed combination of AAPA and Armangau does not render Claims 1-5, 7, 9-11, 14, 17 and 19-20 unpatentable because that combination does not result in a memory controller which moves virtual memory pages from an old page mapping to a new virtual superpage mapping, and can respond immediately to memory accesses using the new virtual superpage mapping even while still copying old physical memory pages to new physical memory pages corresponding to the new virtual superpage mapping. In setting forth the §103(a) rejections, the Office Action proposes many flawed analogies in comparing both AAPA and the system of Armangau to the present invention. All of the following points regarding the rejection of Claims 1-5, 7, 9-11, 14, 17 and 19-20 were previously

raised by Appellants in their earlier-filed Appeal Brief in this case dated March 5, 2008, at pages 5-8.

The first error in the Office Action is the assertion that a "processor" is the same as a "memory controller". The Office Action argues that the admitted prior art discloses the step of instructing a memory controller to move a plurality of virtual memory pages, referring to page 5, lines 16-17 of the specification which states that the operating system (OS) "uses the processor to copy data …." The Office Action explicitly asserts at page 3 that "the 'processor' is analogous to the 'memory controller.'" This assertion is incorrect. A processor of a computer system is not the same thing as a memory controller, and one skilled in the art of memory subsystems would not consider these two different components to be analogous. The prior art computer system shown in Appellants' Figure 1 helps illustrate this point. The processor is reference numeral 22 located within the processing unit, but the memory controller is part of the system memory which is reference numeral 16. Page 10, lines 4-5, of Appellants' specification explains that the memory subsystem includes a memory controller and system memory. Appellants' Figure 4 illustrates the memory controller (42) separately from the processing unit (56) which includes the processor. Every computer scientist understands that the processing unit of a data processing system is not a part of the

memory subsystem, and there is absolutely no objective basis to support such an analogy.

A processor handles generalized instructions such as arithmetic operations (floating-point or fixed-point) using various registers, as well as load/store operations, to access both memory systems and I/O devices. In contrast, a memory controller is limited to managing its associated memory array(s), and a memory controller is responsive to memory data read and memory data write instructions from a processor; indeed the hardware component that sends the instruction to the memory controller in Appellants invention is a processor. These two components have different functionalities and are clearly not interchangeable. The Office Action thus disregards the plain meaning of the explicitly recited term "memory controller" particularly in the claimed context of a data processing system. This interpretation of the term "memory controller" in regard to Claim 1 is also inconsistent with the interpretation given in regard to Claim 14 at page 7 of the Office Action which refers to the storage controller of Armangau as being analogous to a memory controller, while distinguishing the memory controller from the processing unit of Armangau.

The second error in the Office Action is the implication that the prior art software procedure for superpage remapping is analogous to Appellant's claimed hardware-based method. In rejecting Claim 1 the Office Action relies on the

description at page 5, lines 13-17 of Appellants' specification. According to that prior art technique, the OS (software) figures out what instructions are necessary to implement a new virtual superpage mapping, and dispatches that series of instructions to the processor which then carries out those instructions to copy pages to new locations in physical memory corresponding to the virtual superpage. The specification even notes at page 5, line 11-12, that this "solution resorts to software-directed memory copying." Software-based page migration has many disadvantages, among them the extra overhead necessary to direct the copying which interferes with processing performance. The OS must first utilize processor resources to derive appropriate copy instructions, and then further keeps the processor busy carrying out those instructions. More significantly (and as discussed below with regard to Armangau), the new memory addresses cannot be used until after all of the copying is completed, at which time the page table entries are coalesced into the superpage. During this wait, the application continues to suffer from poor translation-lookaside buffer (TLB) behavior (see page 5, lines 18-20).

In contrast, the present invention is achieved through hardware controls without the need for dispatching such a series of instructions to the processor, thereby reducing processing overhead. The hardware support is achieved using the memory controller which directly supervises the page migration, hence the title of

the present invention ("HARDWARE SUPPORT FOR SUPERPAGE COALESCING"). The memory controller receives simplified instructions which define the set of pages to be copied, and a state engine within the memory controller uses direct memory access to carry out copying (see page 10, line 25 through page 11, line 17). The OS can use the new mappings right away because the memory controller maintains a temporary mapping from the new physical pages back to the old physical pages until the new physical pages are brought up to date. As a result, the application TLB behavior improves immediately. Since the new addresses can be used even before copying is complete, the copying can further be carried out opportunistically, imparting even more operational efficiency to the memory subsystem. The means for carrying out this hardware-based control is explicitly recited in Claims 1, 7 and 14, viz., the "memory controller" whose relevant hardware structure is shown in Appellants' Figure 3. Thus the distinction between software supervision of page remapping and hardware-based control is significant, and these two techniques are accordingly not analogous.

The third error in the Office Action is the assertion that Armangau accesses a virtual superpage using a new mapping while the memory controller is still copying pages. The Office Action acknowledges that the admitted prior art does not disclose accessing the virtual superpage using the new mapping while the memory controller is still copying pages (pages 3, 7 of the Office Action), or using

a mapping table for this function (page 5 of the Office Action); however, it is clear that Armangau has nothing to do with page migration or the creation of a virtual superpage and never discusses these concepts by any terminology, and further does not use any supposed new page mappings while copying is still underway. Armangau is directed to a technique for backing up and restoring data in case of a storage system failure. Accordingly, Armangau is not analogous art because it relates to backup systems rather than memory mapping systems, and the backup command of Armangau has no analogy to remapping instructions. The backup/restore process of Armangau involves copying data from location A to location B then, at a later time when recovery becomes necessary, re-copying data back from location B to location A. The "snapshot copy volume" of Armangau mentioned in the Office Action is merely this backup version of data. The snapshot copy volume is not a new page mapping, that is, it does not take a set of old pages and coalesce them into a new superpage with different (more efficient) page addresses; it is just a reproduction of the data. The Office Action thus again disregards the meaning of the term "new virtual superpage mapping" which is constructed from "a plurality of virtual memory pages" as explicitly recited in Claim 1, and disregards similar terms recited in Claims 7 and 14.

Moreover, if any process is required to access data in the production volume of Armangau, it does so using the original memory locations (A) for the data, not

by accessing the backup location (B). The Office Action incorrectly asserts that "the read/write access during snapshot maintenance is analogous to access operations while copying," because any such read/write access will use the original memory locations (the production set), not the ostensible new mapping which corresponds to the snapshot volume according to the Office Action interpretation (page 3). The specific relevant language of Armangau at column 2, lines 16-18, states that "the production data set is accessible to a host processor for read/write access during maintenance of the snapshot copy." Therefore, the accesses are made to "the production data set" only, and the text of Armangau never says that processor instructions use addresses of the snapshot volume. The Office Action analogy of the snapshot volume to a new virtual superpage mapping thus fails for many reasons.

Each of independent Claims 1, 7 and 14 recite memory controller-managed copying, and accessing a virtual superpage using new page mappings before copying has been completed. Since the proposed combination of the admitted prior art and Armangau still fails to result in these features it accordingly cannot render the present invention unpatentable, and the Office Action fails to make a *prima facie* case of obviousness. The foregoing arguments apply equally to dependent Claims 2-6, 8-13 and 15-20. These differences between the present

invention and Armangau are also reflected in the dependent claims as explained below.

(i) Claims 3 and 9

Claims 3 and 9 specify that a read operation for an address of the new page mapping which is currently being copied is actually handled using the corresponding address of the old page mapping (described in Appellants' specification at page 14, lines 14-17). In rejecting these claims the Office Action refers solely to Armangau column 2, lines 16-18. That text of Armangau is quoted above, and says only that the production set is accessible during maintenance of the snapshot copy. There is no discussion about a read operation, or about using an old page mapping corresponding to a read address. This feature of Appellants' invention allows the read access to complete immediately without checking to see if the copying to the new virtual superpage is complete, and likewise without having to wait for the copying to complete. This point was previously raised by Appellants in their earlier-filed Appeal Brief in this case dated March 5, 2008, at page 9.

Rejection of Claims 6 and 16 under §103(a)

The foregoing arguments apply to the rejection of Claims 6 and 16 since those claims depend respectively from Claims 1 and 14, and the rejection is based primarily on the flawed combination of AAPA and Armangau. Claims 6 and 16 specify updating an entry in a cache memory by modifying an address tag according to the new page mapping (illustrated in Appellants' Figure 5). In rejecting these claims, the Office Action refers to Christie column 4, lines 40-50. That text of Christie refers to "virtual tags" which are invalidated by de-asserting "valid bits" in a "valid register" (ref. numeral 114). Christie distinguishes between these valid bits and virtual addresses. The virtual addresses separate from the valid bits, and are stored in the virtual tag register (ref. numeral 110). Accordingly, de-assertion of a valid bit in the valid register does not remap an address which is in the virtual tag register. This feature of Appellants' invention allows the superpage construction to complete without having to flush the contents of any affected cache memory. These points concerning Christie were not previously raised by Appellants since Christie was first cited in the June 12, 2008, Office Action which is the subject of this appeal.

Rejection of Claim 8 under §103(a)

The foregoing arguments apply to the rejection of Claim 8 since that claim depends from Claim 7, and the rejection is based primarily on the flawed combination of AAPA and Armangau. Romer is relied on only for the construction of a mapping table having paired entries. Romer does not disclose or suggest using new virtual superpage mappings while copying of the underlying memory pages is still ongoing. This point was previously raised by Appellants in their earlier-filed Appeal Brief in this case dated March 5, 2008, at page 11.

Rejection of Claims 12-13 under §103(a)

The foregoing arguments apply to the rejection of Claims 12-13 since those claims depends directly or indirectly from Claim 7, and the rejection is based primarily on the flawed combination of AAPA and Armangau. Claims 12 and 13 refer to the state engine within the memory access device that reads the matching old and new page addresses, and the use of a direct memory access engine that carries out the copying, controlled by the state engine (illustrated in Appellants' Figure 3). In rejecting these claims, the Office Action asserts that the mapping module of Waldspurger is analogous to the state engine. The mapping module of Waldspurger is a component of the operating system and is accordingly software, not hardware. The mapping module is disclosed as being part of a "manager"

which is additionally described as being in an intermediate software layer. In contrast, the state machine recited in Claim 12 is part of the memory subsystem of the present invention, in particular the memory controller, which is hardware. This point was not previously raised by Appellants since Waldspurger was first cited in the June 12, 2008, Office Action which is the subject of this appeal.

(i) Claim 13

The Office Action asserts that the memory management unit (MMU) of Waldspurger is analogous to a DMA engine. Claim 13 specifies that the DMA engine carries out actual copying of the memory pages. In addressing this recitation the Office Action relies on Waldspurger column 7, line 67 through column 8, line 3, which states in relevant part that "MMU 116 then performs the actual translation of VPNs to PPNs using these mappings." This text does not state that the MMU performs "copying." Address translation is not the same as nor analogous to copying. Translation of a virtual page number to a physical page number does not involve copying of the page. These features of Appellants' invention allow the copying to be carried out opportunistically as noted in Appellants' specification at page 11, lines 11-14. This point was not previously raised by Appellants since Waldspurger was first cited in the June 12, 2008, Office Action which is the subject of this appeal.

Rejection of Claim 15 under §103(a)

The foregoing arguments apply to the rejection of Claim 15 since that claim depends from Claim 14, and the rejection is based primarily on the flawed combination of AAPA and Armangau. Claim 15 specifies that the processing unit includes a processor core having a translation lookaside buffer (TLB) whose entries are updated prior to completion of page copying. The Office Action asserts that Talluri teaches such updating of the TLB entries. In rejecting Claim 15 the Office Action relies on page 3 of Talluri, right column, second full paragraph. That text of Talluri merely describes copying memory pages, and thereafter updating the TLBs. The text does not say anything about TLBs being updated prior to completion of copying. As noted above, this feature allows the application TLB behavior to improve immediately. This point was not previously raised by Appellants since Talluri was first cited in the June 12, 2008, Office Action which is the subject of this appeal.

Rejection of Claim 18 under §103(a)

The foregoing arguments also apply to the rejection of Claim 18 since that claim depends from Claim 16 and indirectly depends from Claim 14, and the rejection is based primarily on the flawed combination of AAPA and Armangau.

Arimilli is relied on only for the cache supposedly relocating a cache entry. Arimilli does not disclose or suggest using new virtual superpage mappings while copying of the underlying memory pages is still ongoing. The text of Arimilli noted in the Office Action describes the assignment of addresses to congruence classes by switching address bits. However, this assignment of addresses is arbitrary, and so cannot be considered to be modified to correspond to a new memory location in the new page address. Moreover, Arimilli does not actually teach moving a cache entry from one location to another. A memory block in Arimilli stays in one cache location until it is evicted. These points were not previously raised by Appellants since Arimilli was first cited in the June 12, 2008, Office Action which is the subject of this appeal.

Conclusion

The Office Action relies on invalid comparisons to the prior art and on inconsistent interpretations of the prior art. Moreover, the primary proposed combination in support of the grounds of rejection still does not result in Appellants' invention as recited in each of the independent Claims 1, 7 and 14 since the process of Armangau does not use any analog of a new virtual superpage mapping to accessing pages while copying is still underway. Accordingly, the Office Action fails to make out a *prima facie* case for any of the rejections.

Appellants have made a diligent effort to advance the prosecution of this application by pointing out the manifest errors in the Office Action rejection and explaining with specificity how the claims as presented patentably define the invention over the prior art of record. In view of the arguments set forth herein, the application is believed to be in condition for allowance and Appellants respectfully request that the Board of Appeals remand this case to the Examiner with instructions to allow the claims under appeal.

Respectfully submitted,

/Jack V. Musgrove/

Jack V. Musgrove
Attorney for Appellant(s)
Reg. No. 31,986
2911 Briona Wood Lane
Cedar Park, Texas
Telephone: 512-689-6116
Facsimile: 512-918-9536
Email: patentlaw@austin.rr.com

APPENDIX—CLAIMS

The following claims are pending in this application and are involved in this appeal:

1. (rejected) A method of assigning virtual memory to physical memory in a data processing system, comprising the steps of:

allocating a set of physical memory pages of the data processing system for a new virtual superpage mapping;

instructing a memory controller of the data processing system to move a plurality of virtual memory pages corresponding to an old page mapping to the set of physical memory pages corresponding to the new virtual superpage mapping; and

accessing at least one of the virtual memory pages using the new virtual superpage mapping while the memory controller is copying old physical memory pages to new physical memory pages.

2. (rejected) The method of Claim 1 wherein said allocating step allocates a contiguous set of physical memory pages.

3. (rejected) The method of Claim 1 wherein said accessing step includes the step of directing a read operation for an address of the new page mapping which is currently being copied to a corresponding address of an old page mapping.

4. (rejected) The method of Claim 1 wherein said accessing step includes the step of directing a write operation for an address of the new page mapping which is currently being copied to both the address of the new page mapping and a corresponding address of an old page mapping.

5. (rejected) The method of Claim 1 wherein said accessing step includes the step of directing a write operation for an address of the new page mapping which has not yet been copied to a corresponding address of an old page mapping.

6. (rejected) The method of Claim 1, further comprising the step of updating an entry in a cache memory of the data processing system which corresponds to a memory location in the virtual memory page, by modifying an address tag of the cache entry according to the new page mapping.

7. (rejected) A memory controller comprising:

an input for receiving remapping instructions for a virtual superpage;

a mapping table which temporarily stores entries of old page addresses and corresponding new page addresses associated with the page remapping instructions; and

a memory access device which directs the copying of memory pages from the old page addresses to the new page addresses while handling access operations which use the new page addresses, and releases the entries in said mapping table as copying for each entry is completed.

8. (rejected) The memory controller of Claim 7 wherein said mapping table has 32 slots for receiving corresponding pairs of the old page addresses and new page addresses.

9. (rejected) The memory controller of Claim 7 wherein said memory access device directs a read operation for a new page address which is currently being copied to a corresponding old page address.

10. (rejected) The memory controller of Claim 7 wherein said memory access device directs a write operation for a new page address which is currently being copied to both the new page address and a corresponding old page address.

11. (rejected) The memory controller of Claim 7 wherein said memory access device directs a write operation for a new page address which has not yet been copied to a corresponding old page address.

12. (rejected) The memory controller of Claim 7 wherein said memory access device includes a state engine which sequentially reads the paired old and new pages addresses in said mapping table.

13. (rejected) The memory controller of Claim 12 wherein said memory access device further includes a direct memory access (DMA) engine controlled by said state engine which carries out actual copying of the memory pages.

14. (rejected) A computer system comprising:

a processing unit;

an interconnect bus connected to said processing unit;

a memory array; and

a memory controller connected to said interconnect bus and said memory array, wherein said memory controller copies memory pages from old page addresses to new page addresses according to a new virtual superpage mapping while handling access operations which use the new page addresses and while said processing unit carries out program instructions using the new page addresses.

15. (rejected) The computer system of Claim 14 wherein:

said processing unit includes a processor core having a translation lookaside buffer (TLB) whose entries keep track of current virtual-to-physical memory address assignments; and

said TLB entries are updated for the new page addresses prior to completion of copying of the memory pages by the memory controller.

16. (rejected) The computer system of Claim 14 wherein:

said processing unit has a processor core and an associated cache; and

said cache modifies an address tag of a cache entry which corresponds to a memory location in the new page addresses.

17. (rejected) The computer system of Claim 16 wherein said cache modifies the address tag of the cache entry in response to a determination that the cache entry contains a valid value which is not present elsewhere in the system.

18. (rejected) The computer system of Claim 16 wherein said cache further relocates the cache entry based on a changed congruence class for the modified address tag.

19. (rejected) The computer system of Claim 14 wherein said memory controller includes:

a mapping table which temporarily stores entries of old page addresses and corresponding new page addresses; and

a memory access device which directs the copying of the memory pages from the old page addresses to the new page addresses and releases the entries in said mapping table as copying for each entry is completed.

20. (rejected) The computer system of Claim 14 wherein said processing unit, said interconnect bus, said memory array and said memory controller are all part of a first processing cluster, and further comprising a network interface which allows said first processing cluster to communicate with a second processing cluster, said memory controller having at least one pointer for a new page address which maps to a physical memory location in said second processing cluster.

APPENDIX—CLAIM SUPPORT AND DRAWING ANALYSIS

Independent Claims 1, 7 and 14 stand rejected. Claims 3, 6, 8, 9, 12-13, 15, 16 and 18 are separately argued. These claims are set out below with annotations indicating in bold face between braces { } where a claim feature is described in the specification by reference to page and line numbers, and to the drawings:

Claim 1

A method of assigning virtual memory to physical memory in a data processing system, comprising the steps of:

allocating a set of physical memory pages of the data processing system for a new virtual superpage mapping **{p. 14, ll. 2-7; Fig. 6, ref. numeral 92}**;

instructing a memory controller **{p. 10, ll. 21-27; Fig. 3, ref. numeral 42}** of the data processing system **{p. 11, ll. 18-27; Fig. 4, ref. numeral 52}** to move a plurality of virtual memory pages corresponding to an old page mapping to the set of physical memory pages corresponding to the new virtual superpage mapping **{p. 14, ll. 6-8 ; Fig. 6, ref. numeral 94}**; and

accessing at least one of the virtual memory pages using the new virtual superpage mapping **{p. 14, ll. 8-9; Fig. 6, ref. numeral 96}** while the memory controller is copying old physical memory pages to new physical memory pages **{p. 14, ll. 13-20; Fig. 6, ref. numeral 104}**.

Claim 3

The method of Claim 1 wherein said accessing step includes the step of directing a read operation for an address of the new page mapping which is currently being copied to a corresponding address of an old page mapping **{p. 14, ll. 14-17; Fig. 6, ref. numeral 104}**.

Claim 6

The method of Claim 1, further comprising the step of updating an entry in a cache memory **{p. 12, ll. 20-24; Fig. 5, ref. numeral 70}** of the data processing system which corresponds to a memory location in the virtual memory page, by modifying an address tag of the cache entry according to the new page mapping **{p. 13, ll. 1-9; Fig. 5, ref. numeral 76}**.

Claim 7

A memory controller comprising:

an input for receiving remapping instructions for a virtual superpage **{p. 10, ll. 21-27; Fig. 3, unnumbered horizontal arrow on left}**;

a mapping table which temporarily stores entries of old page addresses and corresponding new page addresses associated with the page remapping instructions **{p. 10, ll. 27 through p. 11, l. 6; Fig. 3, ref. numeral 46}**; and

a memory access device which directs the copying of memory pages from the old page addresses to the new page addresses while handling access operations which use the new page addresses, and releases the entries in said mapping table as copying for each entry is completed **{p. 11, ll. 7-14; Fig. 3, ref. numerals 48, 50}**.

Claim 8

The memory controller of Claim 7 wherein said mapping table has 32 slots for receiving corresponding pairs of the old page addresses and new page addresses **{p. 11, ll. 4-6; Fig. 3, ref. numeral 46}**.

Claim 9

The memory controller of Claim 7 wherein said memory access device directs a read operation for a new page address which is currently being copied to a corresponding old page address **{p. 14, ll. 14-17; Fig. 6, ref. numeral 104}**.

Claim 12

The memory controller of Claim 7 wherein said memory access device includes a state engine which sequentially reads the paired old and new pages addresses in said mapping table **{p. 11, ll. 7-9; Fig. 3, ref. numeral 48}**.

Claim 13

The memory controller of Claim 12 wherein said memory access device further includes a direct memory access (DMA) engine controlled by said state engine which carries out actual copying of the memory pages **{p. 11, ll. 8-11; Fig. 3, ref. numeral 50}**.

Claim 14

A computer system comprising:

a processing unit **{p. 11, ll. 20-23; Fig. 4, ref. numeral 56}**;

an interconnect bus connected to said processing unit **{p. 11, ll. 22-25; Fig. 4, ref. numeral 58}**;

a memory array **{p. 10, ll. 5; p. 12, ll. 1-8; Fig. 4, ref. numeral 44}**; and

a memory controller connected to said interconnect bus and said memory array, wherein said memory controller copies memory pages from old page addresses to new page addresses according to a new virtual superpage mapping while handling access operations which use the new page addresses and while said processing unit carries out program instructions using the new page addresses **{p. 10, ll. 21 through p. 11, ll. 17; Fig. 4, ref. numeral 42}**.

Claim 15

The computer system of Claim 14 wherein:

said processing unit includes a processor core having a translation lookaside buffer (TLB) whose entries keep track of current virtual-to-physical memory address assignments **{p. 4, ll. 19-21; Fig. 4, ref. numeral 56; Fig. 1, ref. numeral 29}**; and

said TLB entries are updated for the new page addresses prior to completion of copying of the memory pages by the memory controller **{p. 14, ll. 13-17; Fig. 6, ref. numeral 102}**.

Claim 16

The computer system of Claim 14 wherein:

said processing unit has a processor core and an associated cache **{p. 11, ll. 22-25; Fig. 4, ref. numeral 56}**; and

said cache modifies an address tag of a cache entry which corresponds to a memory location in the new page addresses **{p. 13, ll. 1-9; Fig. 5, ref. numeral 76}**.

Claim 18

The computer system of Claim 16 wherein said cache further relocates the cache entry based on a changed congruence class for the modified address tag **{p. 13, ll. 14-20; Fig. 5, ref. numeral 80}**.

APPENDIX—MEANS OR STEP PLUS FUNCTION ANALYSIS

There are no claims with means or step plus function involved in this appeal.

APPENDIX—EVIDENCE

No evidence is being submitted with this appeal.

APPENDIX—RELATED CASES

There are no appeals, interferences or judicial proceedings related to this appeal.